
Quality Assurance Tests:
- **Debugging** is a process of finding out where something went wrong and correcting the code to eliminate the errors or bugs that cause unexpected results. A software debugging system can provide tools for finding errors in programs and correcting them.
- **Kinds of errors**:
  1. **Language (syntax) errors** are result of incorrectly constructed code, such as an incorrectly typed keyword or punctuations. They are easiest error to be detected on simple running system
  2. **Run-time errors** are detected on running, when a statement attempts an operation that is impossible to carry out. E.g.: if program tries to access a non-exist file or object, it occurs
  3. **Logic errors** occur when expected output is not formed. They can detected only by testing the code and analyzing the results performed by intended codes
- The elimination of syntactical bug is the process of debugging, whereas detection and elimination of logical bug is the process of testing. Quality assurance testing can be divided into two major categories: error–based testing and scenario–based testing
- **Error–based testing** techniques search a given class’s method for particular clues of interests, then describe how these clues should be tested. E.g: Boundary condition testing
- **Scenario–based testing** also called usage–based testing, concentrates on capturing use –cases. Then it traces user’s task, performing them with their variants as tests. It can identify interaction bugs. These are more complex tests tend to exercise multiple subsystems in a single test covering higher visibility system interaction bugs
- **S/w testing** is one element of a broader topic that is often referred to as verification and validation (V&V). **Verification** refers to set of activities that ensure that software correctly implements a specific function. **Validation** refers to different set of activities that ensure that s/w that hs been built is traceable to customer requirements

Testing Strategies:
- The objective of s/w testing is to uncover errors. The various testing strategies constitutes –
  1. **Unit Testing** – Black Box testing, White black testing
  2. **Integration Testing** – Top–down testing, Bottom–up testing, Regression testing
  3. **Validation Testing** – Alpha test, Beta test and
  4. **System Testing** – Recovery testing, Security testing, Stress testing, Performance testing
- **Tom Gilb** argues following issues for successful s/w testing strategy is to be implemented:
  i. Specify product requirements in a quantifiable manner long before testing commences
  ii. State testing objectives explicitly
  iii. Understand the users of s/w and develop a profile for each user category
  iv. Develop a testing plan that emphasizes “rapid cycle testing”
  v. Build “robust” s/w that is designed to test itself
  vi. Use effective formal technical reviews as a filter prior to testing
  vii. Conduct formal technical reviews to assess the test strategy and test cases themselves
  viii. Develop a continuous improvement approach for testing process
- **Unit and integration tests** concentrate on functional verification of a module and incorporation of modules into a program structure. **Validation testing** demonstrates traceability to s/w requirements and system testing validates s/w once it has been incorporated into a larger system
- **Unit test focuses** verification effort on smallest unit of s/w design the module. It constitutes two inner types of testing – White box testing and Black box testing
- **Black box testing**: The concept of black box is used to represent a system that’s inside working are not available for inspection. In black box testing, we try various inputs and examine resulting output though which we learn what the box does nor how conversion takes place
- **White box testing**: White box testing assumes that specific logic important and must be tested to guarantee system’s proper functioning. One form of white box testing called path testing, makes certain that each path in a object’s method is executed at least once during testing & is of types
  1. Statement testing coverage: Its aim is to test every statement in object’s method at least once
  2. Branch testing coverage: Idea is to perform enough tests ensuring all branches are perfect
- **Integration Testing**: It is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. The object is to take unit tested modules and build a program structure that has been dictated by design. It has again 3 testing patterns: Top–down testing, Bottom – up testing and regression testing
Object orientation on Testing:

- The *impact of an object orientation on testing* can be summarized as follows:
  1. Some types of errors could become less plausible (not worth for testing)
  2. Some types of errors could become more plausible (worth testing for now)
  3. Some new types of errors might appear

- *Impact of Inheritance in Testing*: If designers do not follow OOD guidelines especially, if test is done incrementally, it will lead with objects that are extremely hard to debug and maintain

- *Reusability of Tests*: Marick argues that simpler is a test, more likely it is to be reusable in sub classes. The models developed for analysis & design should be used for testing as well

Test Cases:

- *Myers* describes the object of testing as follows:
  - Testing is process of executing a program with the intent of finding errors
  - A good test case is one that has a high probability of detecting an as–yet undiscovered error
  - A successful test case is one that detects as–yet undiscovered error

- *Guidelines for developing Quality Assurance Test Cases*: Freedman & Thomas have developed guidelines that have been adapted for the Unified Approach
  - Describe which feature or service (external or internal), test attempts to cover
  - If test case is based on use case it must refer to use–case name and write test plan for that piece
  - Specify what to test on which method along with test feature and expected action
  - Test normal use of the object’s methods
  - Test abnormal but reasonable use of the object’s methods
  - Test abnormal and unreasonable use of object’s methods
  - Test boundary conditions of number of parameters or input set of objects
Test object’s interactions & message sent among them with assist of sequence diagram
On doing revision, document the cases so they become the starting basis for follow–up test
Attempt to reach agreement on answers of what–if questions and repeat process until stabilized
The internal quality of s/w such as its reusability and extendibility should be assessed as well

Example: Testing a File Open feature, we specify the result as follows:
1. Drop down the File menu and select Open
2. Try opening following type of files
   * A file that is there (should work)
   * A file that is not there (should get an error message)
   * A file name with international characters (should work)
   * A file type that the program does not open (should get a message or conversion dialog box)

Test Plan : -
A test is developed to detect and identify potential problems before delivering the s/w to its users. The test plan need not be very large; in fact, devoting too much time to the plan can be counter productive. The following steps are needed to create a plan
1. Objectives of test: Create objectives of test and describe how to achieve them
2. Development of test case: Develop test data, I/O based on domain of data & expected behavior
3. Test analysis: It involves examination of test O/p and documentation of test results
All passed tests should be repeated with revised program called regression testing. Most s/w companies use beta testing, a popular inexpensive, effective way to test s/w and alpha testing

Guidelines for developing Test Plans: Thomas stated following guidelines for writing test plans
- Specific appearance or format of test plan must include more details about test
- It should contain a schedule and a list of required resources including number of peoples & time
- Document every type of test planned with level of detail driven by several factors
- A configuration control system provides a way of tracking changes to code should exist
- Try to develop a habit of routinely bring test plan sync with product or product specification
- At end of each moth or as reach each milestone, take time to complete routine updates

Continuous Testing: -
A common practice among developers is to turn over applications to a quality assurance (QA) group for testing only after development is completed. As it is not initial plan, it is time consuming and on identification of design weakness whole process must be repeated.
In order to overcome time constraint, testing must take place on a continuous basis & refining cycle must continue throughout the development process until fully satisfied with results
The use cases and usage scenarios can become test scenarios & will drive test plans. The steps to successful testing are
- Develop an internal infrastructure to support continuous testing
- Look for leaders who will commit to and own the process
- Measure & document the findings in a defect recording system
- Publicize improvements as they are made and let people know what they are doing better

Myer’s Debugging principles: -
Bug Locating principles: * Think if you reach an impasse, sleep on it
   * If the impasse remains, describe the problem to some one else
   * Use debugging tools (slightly different from Myer’s suggestion)
   * Experimentation should be done as a last resort (slightly different from Myer’s)
Debugging principles: * When where is one bug, there is likely to be another
   * Fix the error, not just the symptom of it
   * Probability of solution being correct drops as size of prg., increases
   * Beware of possibility that an error correction will create a new error (this is less of a problem in object – oriented environment)

System Usability: -
Usability should be a subset of s/w quality characteristics. This means usability must be place at same level as other characteristics such as reliability, correctness and maintainability
Usability testing deal with how well the interface of s/w fits the use cases, which are reflections of users’ needs and expectations. To ensure user satisfaction, we must measure it throughout system development with user satisfaction tests. It forms comm., vehicle between designers and end users

Usability Testing: -
ISO defines usability as effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in particular environments. It requires
i. Defining tasks. What are the tasks?
ii. Defining users. Who are the users?
iii. A means for measuring effectiveness, efficiency and satisfaction. How do we measure usability?

- **Usability testing** measures the ease of use as well as degree of comfort and satisfaction users have with the s/w. Usability test cases begin with identification of use cases that can specify the target audience, tasks and test goals. When designing test, focus on use cases or tasks
- The **main advantage** is that all design traces directly back to user requirements. Use cases and usage scenarios can become test scenarios; and therefore, the use case will drive usability, user satisfaction & quality assurance test cases

- **Guidelines for developing usability testing**
  - Usability testing should include all of a s/w’s components
  - Usability testing need not be very expensive or elaborate
  - All tests need not involve many subjects. Typically, quick, iterative tests with small, well – targeted sample of 6 – 10 participants can identify 80 – 90 percent of most design problems
  - User’s experience also as part of s/w usability. 80 – 90 percent of most design problems can be studied with target few users of single skill level of users, such as novices or intermediate level
  - Apply usability testing early and often.

- **Recording the Usability Test**
  - A quiet location, free from distractions environment is best for conducting test and intervention yields better results. It is done with test data along with guides or hints around a problem. Always records techniques & search patterns users employ when attempting to work through a difficulty & number and type of hints provided to them

**User Satisfaction Test:**

- **User satisfaction testing** is process of quantifying usability test with some measurable attributes of test, such as functionality, cost, intuitive UI, reliability or ease of use. Usability can be accessed by defining measurable goals such as
  - 95% of users should be able to find how to withdraw money from ATM machine without error and no formal training
  - 70% of all users should experience new function as “clear improvement over the previous one”
  - 90% of consumers should be able to operate VCR within 30 minutes

- The **principle objectives** of user satisfaction test are
  - Act as communication vehicle between designers as well as between users & designers
  - To detect and evaluate changes during the design process
  - To provide a periodic indication of divergence of opinion about current design
  - To enable pinpointing specific areas of dissatisfaction for remedy
  - To provide a clear understanding of just how the completed design is to be evaluated

- **Guidelines for developing user satisfaction test**
  - The format of every user satisfaction test is basically same with contents different for each project. The work must be done with users or clients finding out what attributes should be included in test. Ask users to select limited number (5 to 10) of attributes by which final product can be evaluated.
    - User might select following attributes for customer tracking system: ease of use, functionality, cost, intuitiveness of user interface and reliability. User must his/her judgment to answer each question by selecting number between 1 – 10 with 10 as most favorable & 1 as least
  - An **interesting side effect** of developing user satisfaction tests is that we benefit from it even if test is never administered to anyone; it still provides useful information. Performing test regularly user actively involved in system development. It also helps us stay focused on user’s wishes

**Quality standards – CMM :**

- **Level 1** – Effective analysis – Design Information to projects
- **Level 2** – Reviews & audits
- **Level 3** – Extended procedures & new coding entries
- **Level 4** – 1. Quantitative process management, 2. s/w quality ;management
- **Level 5** – Three different areas 1. Defect prevention – s/w quality management 2. Technology change management – technical basics 3. Process change management – change of programs must not affect work

**Questions:**

- 10 marks
  1. What is a testing? Discuss various testing strategies available?
  2. Write short notes on (a) Testing strategies. (b) Object persistence.